American Journal of Applied Sciences 4 (9): 666-672, 2007
ISSN 1546-9239
© 2007 Science Publications

A Novel High-Speed 54×54 bit Multiplier

[1]Pouya Asadi and [2]Keivan Navi
[1]Islamic Azad University Science and Research Branch,Tehran, Iran
[2]Faculty of Electrical and Computer Engineering Shahid Beheshti University, Tehran, Iran

Abstract: A new 54×54-bit multiplier using high-speed carry-lookahead adder has been fabricated by CMOS technology. This paper presents a self-timed carry-lookahead adder in which the logic complexity was a linear function of n, the number of inputs, and the average computation time was proportional to the logarithm of the logarithm of n. To the best of our knowledge, our adder has the best area-time efficiency. A novel 4-2 compressor, featuring pass-transistor multiplexers, has been developed. The proposed circuits have a speed advantage over conventional CMOS circuits because the number of critical-path gate stages was minimized due to the high logic functionality of pass-transistor multiplexers. The total number of transistors of the proposed multiplier core was 42579 and The multiplication time was 3.4 ns at a 1.3 V power supply.

Keywords: Adder, booth encoder, compressor, CMOS

INTRODUCTION

Multiplication is one of the basic arithmetic operations. In fact 8.72% of all instructions in a typical scientific program are multiplies[1]. Also, multiplication is a long latency operation. In typical processes multiplication takes between 2 and 8 cycles[2]. Consequently, having high speed multipliers is critical for the performance of processors. Processor designers have recognized this, and have devoted considerable silicon area for the design of multipliers[3]. Recent advances in integrated circuit fabrication technology have resulted in both smaller feature sizes and increased die areas[4]. Together, these factors have provided the processor designer the ability to fully implement high-speed floating point multipliers in silicon.

The IEEE 754 floating point standard[5] is the most common floating point representation used in modern microprocessors. This research therefore investigates the area and performance tradeoffs needed in the design of high performance IEEE conforming multipliers. Multiplication can be divided into three steps: generating partial products, summing up all partial products, and adding the remaining two rows of partial products by using a carry propagation adder. Most the above mentioned approaches employ Booth Encoding approach, for the first step because of its ability to cut the number of partial produces rows in half[6]. To find out possible alternatives, it is necessary to decide a logic design style and implementation technology.

MATERIALS AND METHODS

The original Booth's algorithm allows the multiplication operation to skip over any contiguous string of all 1's and all 0's in the multiplier. To date, the most widely adopted technique for partial-product generation in large multipliers (16-bit and above) is the modified Booth's algorithm (MBA). The main attraction of MBA is that instead of generating n partial-products for an n-bit multiplication, it only generates half of that. Many existing designs of PP (Partial Product) generators reply on NMOS pass transistor logic to achieve smaller area and delay[7]. Because of the poor driving capability and degrades high voltage level in NMOS pass transistor logic, careful custom transistor-level design is necessary for circuit robustness. Pull-up PMOS transistors are often needed to force a node to be high[8].

To reduce the number of inverters in pass transistor logic, dual-rail signals are often used. Dual-rail signals increase wiring complexity and switching capacitance significantly. On the other hand, standard CMOS logic style is robust with respect to voltage scaling and transistor sizing. Transmission gates (a pair of NMOS and PMOS transistors) are often used to implement multiplexers, XORs, and flip-flops efficiently. CMOS logic style also has the advantages of generality and ease-of-use as standard cell based technology libraries and logic synthesis techniques are well developed and widely used. It has shown that CMOS logic style is the

Corresponding Author: Pouya Asadi, Islamic Azad University Science & Research branch, Tehran, Iran, Tel: +98-21-66422539

right choice in most cases if low voltage, low power, small delay and small power-delay products are of major concern[9]. For these reasons, we choose CMOS standard cell logic style in this study. For radix-4 recoding, the recoder itself only occupies less than 2% of total cell area and the power consumption is even less than 1% of total power consumption in a 54×54-bit multiplier. But the recoder design determines the complexity of PP generators which occupy over 30% area and consume about 10% power. In addition, different recoders introduce different unbalanced signal propagation delays in the PP generation and reduction logic, which also affects power consumption significantly. Among existing recoding schemes, neg/two/one, neg/pos/two and P2/N2/P1/N1/ZERO have relatively simple recoder and PP generator designs. Which one is actually better depends on how the multiplexers and XORs are implemented in the cell library. When AOI22 (i.e., $\overline{ab+cd}$) is smaller than MUX21 and XOR2, neg/two/one would have smaller area. But "-0" in neg/two/one should be handled in the same way as "+0" for power saving. In the following, we propose several alternative recoding schemes and evaluate their power/area/delay characteristics under an industrial standard cell technology. The signal "zero" in the 4 parallel neg/two/zero recoding helps to avoid unnecessary computations for "-0". But the circuitry of neg/two/zero recoding is more complex than that of neg/two/one recoding. To reduce power, we propose to merge the simplicity of neg/two/one recoding with the power-efficient "-0" handling in neg/two/zero recoding. For parallel recoding, neg/two/one can be improved to become Table 1. The new switching are:

$$neg = y_{2i+1}(y_{2i}y_{2i-1})' \quad (1)$$

$$two = y'_{2i+1}y_{2i}y_{2i-1} + y_{2i+1}y'_{2i}y'_{2i-1} \quad (2)$$

$$one = y_{2i} \oplus y_{2i-1} \quad (3)$$

$$crt = \text{neg} \quad (4)$$

There is no change in the PP generation logic.

Table 1: Parallel neg/two/one with unique zero

y_{2i+1}	y_{2i}	y_{2i-1}	OP	neg	two	one	crt
0	0	0	+0	0	0	0	0
0	0	1	+X	0	0	1	0
0	1	0	+X	0	0	1	0
0	1	1	+2X	0	1	0	0
1	0	0	-2X	1	1	0	1
1	0	1	-X	1	0	1	1
1	1	0	-X	1	0	1	1
1	1	1	-0	0	0	0	0

Table 2: Serial neg/two/one recoding with unique zero

y_{2i+1}	y_{2i}	c_i	OP	neg	two	one	crt	c_{i+1}
0	0	0	+0	0	0	0	0	0
0	0	1	+X	0	0	1	0	0
0	1	0	+X	0	0	1	0	0
0	1	1	+2X	0	1	0	0	0
1	0	0	+2X	1	1	0	1	0
1	0	1	-X	1	0	1	1	1
1	1	0	-X	1	0	1	1	1
1	1	1	-0	0	0	0	0	1

For the highest order action with an unsigned multiplier, the action must be derived with a leading or padded zero. For serial recoding, an alternative to neg/one/zero is also neg/two/one with unique handling of "± 0". The recoding relationship is shown in Table 2, which corresponds to the following expressions:

$$two = y'_{2i+1}y_{2i}c_i + y_{2i+1}y'_{2i}c'_i \quad (5)$$

$$one = y_{2i} \oplus c_i \quad (6)$$

$$neg = y_{2i+1}.one \quad (7)$$

$$crt = \text{neg} \quad (8)$$

$$c_{i+1} = y_{2i+1}(y_{2i} + c_i) \quad (9)$$

When 4-to-1 multiplexers have small and fast design, a scheme with P2, P1, N1 and ZERO control signals can be developed for serial recoding. The PP generation logic is a simple 4-to-1 multiplexer. In this scheme, the carries affect more logic and generate more spurious transitions due to carries propagate throughout the PP generation and PP reduction logic, which may overturn any gain in the PP generator.

We have implemented and tested six existing recoding schemes and three alternative recoding schemes in a 54×54-bit linear array multiplier framework. All implementations are written in gate-level VHDL and mapped into Artisan TSMC 0.13 μm , 1.3-Volt standard-cell library. The mapping objective is set to minimize area as area minimization helpful in power saving for a given design. Power estimation is obtained through full-timing gate-level simulation on the Synopsys Power Compiler platform.

Three test data sets are used in order to catch power features in different application environments. One test data set is random, consisting of pseudo-random data. The second data set is djpeg gathered by tracing the execution of multiplication in a 32-bit JPEG-decoding program djpeg with a typical image input (from Media Bench Suite[10]). Djpeg data have large dynamic range and most data precisions are less than 16-bit. The third data set, special, is generated manually by repeating the following patterns:

00000000000000001011001001110110
11111111111111110100110110001001
11111111111111110100110110001001
00000000000000001011001001110110

This is designed to test the efficiently of unique zero handling. The actual precision of each 32-bit number is 16-bit and the higher 16 bits are all sign bits. For each bit in special, both the static probability of being '1' in random is the same as in special, the power consumption under special is different from the power under random because of the difference in sign extension bits.

The power/area/delay comparison results are listed in Table 3. The values in parentheses are normalized values. P-n21 is parallel neg/two/one recoding. P-n20 is parallel neg/two/zero recoding. P-np2 is parallel neg/pos/two recoding. P-np210 is parallel N2/P2/P1/N1/ZERO recoding. S-n10 is serial neg/one/zero recoding. P-n21-u0 and s-n21-u0 are the proposed parallel and serial neg/two/one recoding schemes with unique handling of "± 0". S-p2n10 is serial P2/P1/N1/ZERO recoding. The power consumption is measured at 40MHz and the power unit is mW. The static power is less than 1 μW in all cases. As static power is irrelevant to the clock frequency and is less than 0.1% of the total power, only dynamic power is used for comparison. The experimental results in the table show that neg/two/one recoding has about 11% less area than other recoding schemes. This is because AOI22 has smaller area than MUX21 and XOR2 in Artisan library. Based on this experiment, we find that serial neg/two/one recoding with unique zero handling is a good choice for low-power linear array multipliers.

Together with the advantage of less unbalanced signal propagation paths, neg/two/one recoding helps reduce 15% power under djpeg data and 8% under special when the design goes from p-n20 to p-n21-u0.

Table 3: Comparison of different radix-4 schemes

	Power(mW)			
Schemes	Random	Djpeg	Special	Delay(ns)
p-n21[1]	26.71	14.30	28.04	12.32
p-n20[11]	26.62	13.59	23.25	12.84
p-n20[2]	25.63	13.52	22.69	12.85
p-np2[9]	24.78	12.67	20.85	12.33
p-p210[12]	23.83	12.24	20.63	12.37
p-n21-u0[8]	25.28	11.50	20.90	12.36
s-n10[13]	25.72	12.13	19.24	13.01
s-p2n10[5]	24.99	13.03	23.95	12.85
s-n21-u0[6]	24.36	12.16	17.21	13.00

The unique handling of "± 0" is quite efficient in saving power. From p-np2 to p-n21-u0, the power saving is 20% under djpeg and 25% under special. For parallel recoding, p-np2, p-np210, and p-n21-u0 have similar power consumptions under three test data sets. S-n21-u0 is even better than p-n21-u0, except for djpeg data. One of the differences between djpeg and the other two data sets is that the probability of a number being negative in djpeg is only 30% while such a probability is 50% in random and special. Therefore, the advantage of unique zero handling is not that significant under djpeg data.

Based of this experiment, we find that serial neg/two/one recoding with unique zero handling is a good choice for low-power linear array multipliers and parallel neg/two/one recoding with unique zero handling is a good candidate for tree multipliers if a similar cell-based design technology is used.

New Compressor: Pass-transistor logic (PTL) was reported as an alternative logic that can enhance the circuit performance[9]. Since PTL can propagate signals using either the source (or drain) and the gate, its high functionality can reduce the number of transistors in critical path. As a PTL based circuit can be constituted of only one type MOS transistor (generally NMOS transistor), it has low node capacitance[10]. Because wider bit multiplier has large partial product reduction tree (PPRT), fast reduction method is the key to the high performance. Partial products may be reduced using one of the various compressors and counters scheme. Many papers report that [4:2] compressor is the optimal leaf cell of PPRT in the trade-off of design simplicity and wiring complexity[4]. In this paper [4:2] compressor is chosen for PPRT using dynamic PTL. Fig. 1 shows the [4:2] compressor that consists of 4 input XOR, AND-OR, OR-AND and 2 input multiplexer. The compressed outputs 'X' and 'Y' may become the inputs of the successive carry propagation adder or another compressor for the final multiplication results. Because the internal output carry ('CO') is not a function of the input carry ('CIN') from the previous bit, internal carries are not propagated to the next weight. Therefore, fast reduction is possible. For preventing incorrect operation of PPRT by the skews of the PPG outputs, the compressor is designed by pre-discharged dynamic PTL while the PPG is designed by pre-charged dynamic. The detailed design is illustrated in Fig. 2.

Wallace's tree, shown in Fig. 3, is constructed from partial product generators, 4-2 compressors, full adders, and half adders[11]. Using the 4-2 compressor simplifies the construction of Wallace's tree. The Wallace's tree consists of only five kinds of blocks and four kinds of wire shifters. The five kinds of blocks include an 8×4 partial-product generator (P), eight 4-2 compressors (8C), six half adders (6H), 15 half adders (15H), and 21 half adders and a full adder (21H + 1F). The four kinds of wire shifters include an 8-b right shifter, an 8-b left shifter, a 16-b right shifter, and a 16-b left shifter. The sum and carry signals of 4-2 compressors are shifted by the wire shifters. In addition, the carry signals are shifted to the left one more bit.

HSPICE simulation is carried out with $0.13\,\mu m$ CMOS model[8]. The thickness of the gate oxide/threshold voltage of PMOS and NMOS are 6.2nm/0.43V and 6.0nm/0.35V respectively. Performance of 2-input multiplexer designed by dynamic PTL and CPL is compare through simulation. Dynamic PTL based multiplexer improves both the power consumption and speed. It can be explained that the dynamic PTL removes overlap current through PMOS and it shows fast evaluation characteristic of dynamic logic. Since dynamic PTL has the merits of both conventional PTL and dynamic logic, high performance is expected without additional restoration circuit. Dynamic PTL outperforms CPL in power and delay. It is verified using HSPICE simulation. PTL is faster by 2.5 times than static CMOS. The power delay product is improved to half. Therefore, dynamic PTL is proved to be high performance circuit design logic. The dynamic PTL holds the merits of fast evaluation characteristics. Moreover, because using pre-charged scheme solves weak logic "high" problem of static PTL, additional level restoration circuit is not needed. Since dynamic PTL has the merits of both conventional PTL and dynamic logic, high performance is expected without additional restoration circuit.

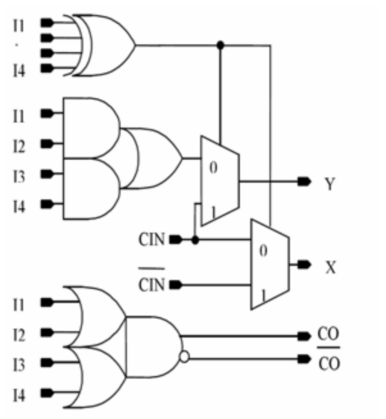

Fig. 1: Schematic diagram of the design [4:2] Compressor

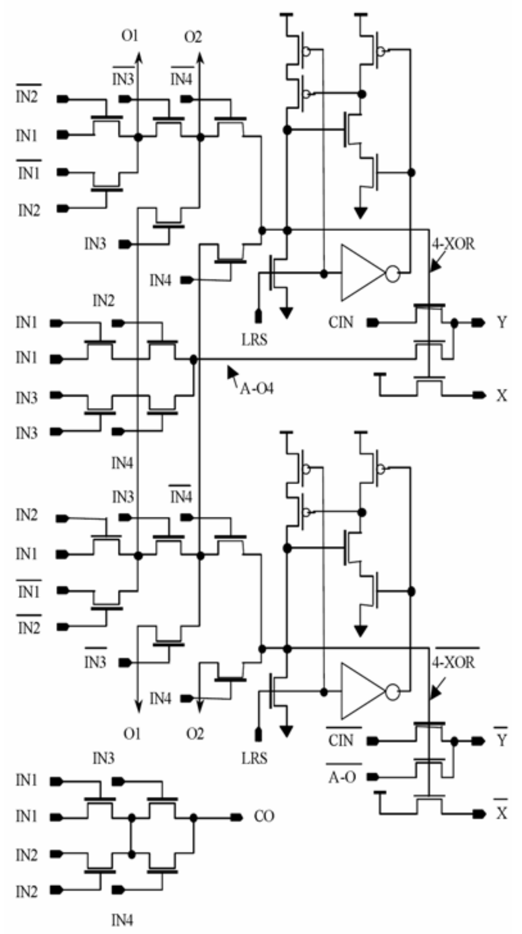

Fig. 2: [4:2] compressor for partial product reduction tree designed by dynamic PTL

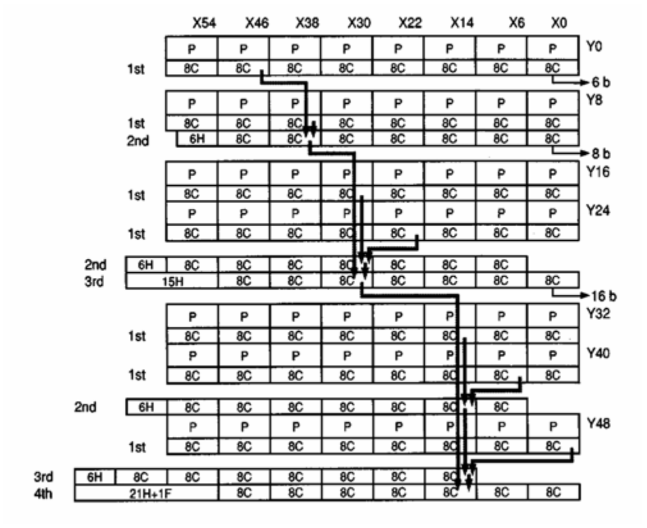

Fig. 3: Construction of Wallace's tree

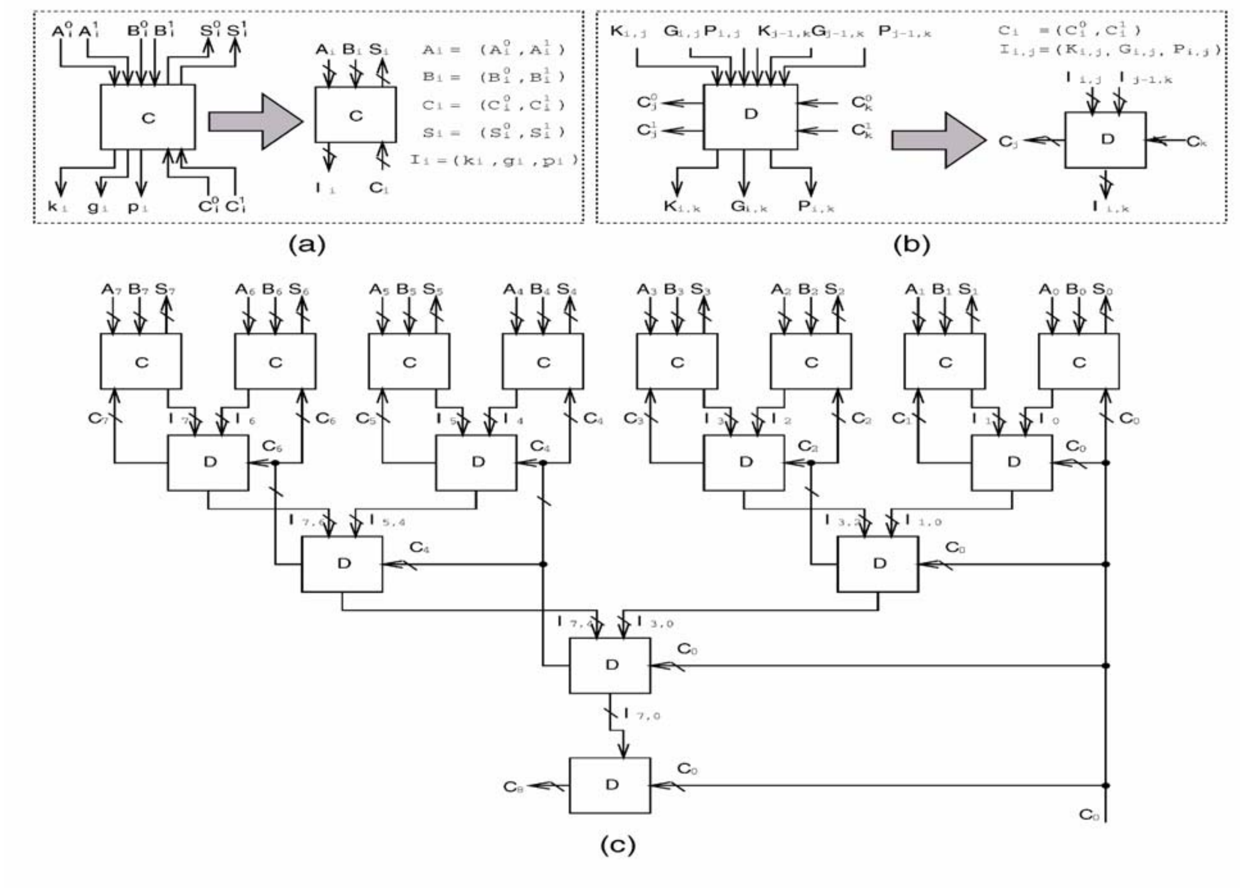

Fig 4: Delay sensitive carry-lookahead adder

Delay-Intensive Carry Look-ahead Adder: Delay-Intensive Carry-Lookahead Adders (DICLA) may be implemented by using dual-rail signaling in input bits, sum bits, and carry bits, and by using one-hot code in the internal signals[14]. A DICLA may be built with two basic modules: C and D, connected in a tree-like structure (Fig. 3)[15]. The equations of the C-module are defined as follows:

$$k_i = \mathrm{A}_i^0 B_i^0 \quad \text{(carry kill)} \tag{10}$$

$$g_i = \mathrm{A}_i^1 B_i^1 \quad \text{(carry generate)} \tag{11}$$

$$p_i = \mathrm{A}_i^0 B_i^0 + A_i^1 B_i^1 \quad \text{(carry propagate)} \tag{12}$$

$$S_i^0 = \mathrm{A}_i^0 B_i^0 C_i^0 + A_i^1 B_i^1 C_i^0 + A_i^0 B_i^1 C_i^1 + A_i^1 B_i^0 C_i^1 \tag{13}$$

$$S_i^1 = \mathrm{A}_i^1 B_i^1 C_i^1 + A_i^1 B_i^0 C_i^0 + A_i^0 B_i^1 C_i^0 + A_i^0 B_i^0 C_i^1 \tag{14}$$

Where i= 0,1,..,n

Note that, in a DICLA, the input data bits and the output data (sum) bits may be propagated at any time in any order. The completion signal, finish, may be generated when needed as follows:

$$finish = (\mathrm{C}_\pi^0 + C_\pi^1)\prod_{i=0}^{i=n}(S_i^0 + S_i^1) \tag{15}$$

For more completion detection circuits for dual-rail self-timed systems[16,17].

The C-module is shown in Fig. 4a. The dual-rail signals on the left hand side of Fig. 4a are grouped as $A_i = (A_i^0, A_i^1)$, $B_i = (B_i^0, B_i^1)$, $C_i = (C_i^0, C_i^1)$, $S_i = (S_i^0, S_i^1)$, and $I_i = (k_i, g_i, p_i)$. The resulting C-module is shown on the right-hand side of Fig. 4a.

The equations for the D-module are defined as follows:

$$P_{i,k} = \mathrm{P}_{i,j} P_{j-1,k} \quad \text{(block carry propagate)} \tag{16}$$

$$K_{i,k} = \mathrm{K}_{i,j} + P_{i,j} K_{j-1,k} \quad \text{(block carry kill)} \tag{17}$$

$$G_{i,k} = G_{i,j} + P_{i,j}G_{j-1,k} \quad \text{(block carry kill)} \qquad (18)$$

$$G_{i,k} = G_{i,j} + G_{j-1,k} \quad \text{(block carry generate)} \qquad (19)$$

$$C_j^0 = K_{j-1,k} + P_{j-1,k}C_k^0 \qquad (20)$$

$$C_j^1 = K_{j-1,k} + P_{j-1,k}C_k^1 \qquad (21)$$

The D-module is shown in Fig. 4b. The signals on the left-hand side of Fig. 4b are grouped as $I_{i,j} = (K_{i,j}, G_{i,j}, P_{i,j})$ and $C_i = (C_i^0, C_i^1)$. The resulting D-module is shown on the right-hand side of Fig. 4b. Initially, all carries (i.e., C_i^0 and C_i^1 for i= 1,2,…,n) and the internal signals (i.e., $K_{i,j}$, $G_{i,j}$ and $P_{i,j}$) are zero because all primary inputs (i.e., A_i^0, A_i^1, B_i^0 and B_i^1 for i=0,1,…,n-1) are zero. During the computation, C_i^0 and C_i^1 will not be both turned on simultaneously and exactly one of the internal signals will be turned on.

The performance of the DICLA may be further improved by some speed-up circuitry. It is obvious that if $A_i = B_i$ (i.e., carry-kill or carry-generate), then the output carry, C_{i+1}, is independent of the input carry, C_i. The tree-like circuit, shown in Fig. 4, does not take full advantage of this feature to speed up the carry computation.

RESULTS AND DISCUSSION

Cells are selected from the standard cell library[18]. The PPRT is optimized by TDM algorithm [19]. Then, the MBE scheme and the optimized PPRT are implemented by using the selected cells. We use VHDL code to describe the design. The code is fed into Synopsys Design Analyzer to insert appropriate buffers. In this step, the only work done be Design-Analyzer is buffer insertion. No cell is changed or removed from our design. The PPRT delay profile, which considers buffer delay, capacitance loading, and driving capability, is obtained from Design-Analyzer. The delay profile is then used as the inputs to the MLCSMA algorithm. In this step, the maximum path delay of multiplexer is 0.41 ns. After the final adder is generated, the PPRT and the final adder are merged into a complete multiplier. Again, the Verilog code of the multiplier is fed into Design-Analyzer to do buffer insertion.

We estimate the delay of PPRT by multiplying the delay profile with unit delay (0.46 ns). Table.4 shows the result and the delay profiles from Design-Analyzer. Clearly, the estimated delay profiles of PPRT are almost the same as those from Design Analyzer. The delay of final adder can also be found. For example, the delay of final adder is 2.3 ns (=6.46 ns - 4.16 ns) for 32×32 multiplier. Because the delay of the 2-input multiplexer used to construct MLCSMA is smaller than one unit delay, the delay of final adder should be normalized with respect to one MUX delay. Therefore, the final adder delay is 2.3/0.41=5.6 MUX delays. The error for the estimated delay is 6.5-5.6 = 0.9 MUX delays. Similarly, the error of the estimated final adder delay for 24×24 multiplier is 0.43 MUX delays. In both cases, the error is smaller than one MUX delay. HSPICE simulation is carried out with $0.18\mu m$ CMOS model. First of all, performance of 2-input multiplexer designed by dynamic PTL and CPL is compared through simulation. Dynamic PTL based multiplexer improves both the power consumption and speed.

Table 4: Comparison between 54×54 multipliers

	This work	[20]	[21]
Technology (μm)	0.13	0.13	0.13
Transistor counts	42579	78800	60779
Multiplication time(ns)	3.4	8.8	4.1
Chip Area(mm^2)	0.89	9.4	1.27
Power Diss(mW/MHz)	1.1	5.4	2.23
PDP(pJ@100MHz)	759	4752	914.3

CONCLUSION

A 54×54-bit CMOS parallel multiplier architecture was proposed in this work to reduce the number of transistors, delay and power consumption. A radix-4 encoding scheme was used to reduce the number of partial products. The total number of transistors of the proposed multiplier core is 42579, which is at least 12.8% smaller than any of the published 54×54-bit CMOS multipliers. The functionality was verified by VHDL and HSPICE simulations. The proposed multiplier was laid out by using a 0.13 μm CMOS process with 5 metal layers, and the chip area of the proposed multiplier core was $0.89\,mm^2$, which was at least 22% smaller than those of other published CMOS multipliers. The performance of time delay and power-consumption was evaluated by simulations using the layout data to take into account of the parasitic effects by junction and interconnect capacitances. The simulations showed that the multiplication time of the proposed multiplier for the worst case input pattern was 3.4 ns. Especially in the power-delay product and the power-delay-area product, the proposed multiplier showed an excellent performance compared to other published multipliers. In this paper, we have shown how to build a high-speed Booth encoder multiplier. By combining the proposed new booth recoder and the modified partial product generation, the multiplier can perform better than the non-Booth based design. For the final adder, a new algorithm that optimizes final adder incrementally is proposed. The proposed algorithm solves final adder problem efficiently for any size and shows performance improvement up to 25 percent to the final adder. This work has been verified by gate level simulation that considers the effects of buffer delay, capacitance loading, and driving capability. The simulation results meet the estimated delay closely. This work has 7.6% reduction in delay and 8.6% improvement in power consumption in compare with other designs.

REFERENCES

1. Olivieri, M., 2004. Design of synchronous and asynchronous variable-latency pipelined multipliers. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 9: 256-262.
2. Satoro, M. R. and M. A. Horowitz, 2004. A pipelined 64×64-bit multiplier. IEEE Journal of Solid-State Circuits, Vol. 24: 642-651.
3. zhuo, L. and V. K. Prasanna, 2007. Scalable and Modular Algorithms for Floating-Point Matrix Multiplication on Reconfigurable Computing Systems. IEEE Transactions on Parallel and Distributed Systems, Vol 18: 433-448
4. Twagiry, H. A. and M. J. Flynn, 2004. Technology scaling effects on multipliers. IEEE Transactions on Computers, Vol. 47: 1201-1215.
5. Cheng, F. and M. Theobald, 2000. Self timed carry-lookahead adders. IEEE Transaction on Computers, vol. 49: 659-672.
6. Hoon, S. and S. Jun, 2002. A Compact radix-64 54×54 CMOS redundant binary parallel multiplier. IEICE Transaction on Electron, vol. 85: 1342-1350.
7. Huang, Z. and M. Ercegovac, 2005. Number representation optimization for low-power multiplier design. IEEE Transaction on Computers, vol. 45: 253-258.
8. Ohkubo, N., M. Suzuki, T. Shinbo, 1995. A 4.4 ns CMOS 54×54-b multiplier using pass-transistor Multiplexer. IEEE Journal of solid-state circuits, VOL 30: 251-257.
9. Kyoung, H. L., 2003. Design of an 8-bit multiplier using dynamic pass transistor logic. IEEE Journal of solid-state circuits, VOL 40: 279-285.
10. Wang, Z. and W. C. Miller, 2005. A new design technique for column compression multipliers. IEEE Transactions on Computers, vol. 44: 962–970.
11. Khabbazian M., T. A. Gulliver, V. K. Bhargava, 2007. A Breakdown Voltage Multiplier for High Voltage Swing Drivers. IEEE Transaction on Computers, Vol 56: 305-313
12. Yano, K., T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi and A. Shimizu, 2000. A 3.8-ns CMOS 16x16-b multiplier using complementary pass-transistor logic. IEEE Journal of Solid-State Circuits, Vol. 25: 254-261.
13. Cheng, F. and M. Theobald, 2005. Design of Synchronous variable-latency pipelined multipliers. IEEE Transaction on Computers, vol. 49: 659-672.
14. Chen, C. and Z. Li, 2006. A low-power CMOS Analog multiplier. IEEE Transactions on Circuits and Systems, Vol. 53:100-104.
15. Gao, S., N. Chabini, D. Al-khalili and P. Langlois, 2007. Optimized realizations of large integer multipliers and squarer using embedded block. IET Computers & Digital Techniques, Vol 1: 9-16.
16. Lee, H. W., J. L. Beutler and A. R. Hawkins, 2006. High Gain Effects for Solid-State Impact-Ionization Multipliers. IEEE Journal of Quantum electronics, Vol42: 471-476.
17. Butas, J., C. S. Choy, J. Povazanec and C. F. Chan., 2005. Asynchronous cross-pipelined multiplier. IEEE Journal of Solid-State Circuits, Vol. 26: 1272-1275.
18. Song P. J. and G. DeMicheli, 2006. Circuit and architecture tradeoffs for high-speed multiplication. IEEE Journal of Solid-State, vol. 26: 1184–1198.
19. Tenca, A. F. and G. Todorov, 2004. High radix design of a scalable modular multiplier. in Cryptographic Hardware and Embedded Systems: 212-219.
20. Kang, J. Y. and J. L. Gaudiot, 2006. A simple high-speed multiplier design. IEEE Transaction on Computers, vol. 55: 1253-1258.
21. Itoh, N., Y. Naumura, H. Makino, Y. Nakase, T.Yoshihara and Y. Horiba, 2006. A 54×54-bit multiplier with rectangular-styled Wallace tree. IEEE journal of Solid-State Circuits, Vol. 36: 249-257